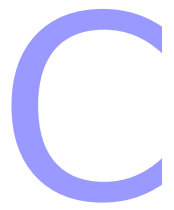
# Süsteemprogrammeerimine keeles C



14 Loeng

# Turbeteemad

- Üldisemalt
- Mõtteid vajalikel teemadel
- C keele spetsiifikast

# Aga enne!

- http://www.youtube.com/watch?feature=player_embedded&v=p5T81yHkHtI

# Allikatest

- http://www.dwheeler.com/secure-programs/
- http://www.ibm.com/developerworks/library/s-buffer-defend.html
- https://www.securecoding.cert.org/confluence/display/seccode/CERT+C+Secure+Coding+Standard
-

# Probleemistik

- Töö häirimine
- Andmete muutmine
- Privileegide suurendamine (privilege escalation)
- Andmete leke

(confidentiality, integrity, availability)

# Miks tekivad turvaprobleemid?

- Oskused
- Ebaturvalised vahendid (C keel)
- Mitme kasutajaga ja paralleelprotsessidega mõtlemine on keeruline
- Laiskus
- Aeg/Raha
- Häid programmeerijaid on vähe
- Kasutajat väga ei huvita
  ... ?

# Paranoia

- Paranoilisus on turvalisuse põhiline alustala
- Tuleb mõelda nagu ründaja
  turvamehe analoogia
- Rünnakutel kehtib eeldus, et kõik vead kasutatakse ideaalsel moel ära
- Kaitsja peab pidevalt kaitses olema, ründajal piisab ühest edukast ründest
  Esimese maailmasõja vastand-analoogia

# Valva sisendit!

- Sisend valetab!

# Käsurida

- Execve() abil saame lisada \0 char-e kohtadesse kus neid ei eelda
- Setuid/setgid programmide probleem

# Keskkonnamuutujad

- Täielikult käivituva programmi kontrolli all
- St IFS muutuja (mis ütleb sh ja bash jaoks milline tähemärk on käsurea argumentide eraldaja)
- Kui programm kasutab system() käsku, palju jama
- Lahendus: keskkond puhastada ja kasutada ainult tarvilikke osi
- (setuid/setgid probleem taas)
- Kasutaja saab LD_PRELOAD abil suvalise .so teegi kaasata (ja ~/.environment muutujas seda muuta)

# Failinimed

- Salakaval . .. ja / märkide olemasolu
- Puhvri ületäitmine PATH_MAX probleemide puhul
- ../*/../*../*/../* tüüpi teenustõke glob() funktsiooniga

# Paroolide küsimine

- Probleem: kasutajalt parooli küsimine nii, et seda ei näidataks ekraanile (kui ta tahvlil praktikumi seletab näiteks)
- Lahendus:

```
#include <unistd.h>
char * = getpasswd(char * prompt)
```

- Võtab ühenduse "päris" terminaliga /dev/tty , kui ei õnnestu, võtab appi stdin ja stderr . Blokeeritakse INTR, QUIT ja SUSP tähemärgid terminalis.
- Terminal flushitakse enne ja pärast parooli kirjutamist

# Mida getpass() teeb?

- Prindib välja argumendi prompt ja tagastab pointeri paroolistringile
- Paneb terminali režiimi, kus tähti välja ei näidata ja taastab esialgse olukorra pärast parooli
- Kuna ei ole *thread-safe*, ning POSIX standardist välja jäetud, ei soovitata kasutada ja võiks pigem ise kirjutada
- Oluliste programmide puhul on hea tava parool kiiresti krüptida ja koht, kus tähed mälus asusid esimesel võimalusel nullidega üle kirjutada.

# Krüptimine: crypt()

- Krüpteerib parooli DES või MD5 algoritmiga:

```
char * crypt(constchar* key, const char* salt);
```

- Usutakse, et kui funktsiooni väljund on antud, siis parim viis algne parool leida, on võimalikud variandid ükshaaval läbi proovida.
  - Õigemini: DES algoritmi puhul ei tasu seda üldse uskuda, MD5 puhul natuke võib
- salt: kui on kahetäheline, valib DES algoritmi, kui soovid MD5, alusta stringi $1$ + kuni 8 tähte, mis lõppevad $ või \0 märgiga

# Salt

- Parooli sissesoolamine on tarvilik selleks, et pelgalt parooli räsi omamisest ei piisaks: ründaja peaks omama parooli räsi kõikide võimalike soolamiste puhuks.
- Väljund on salt + $ (kui algselt polnud) + räsi
- Parooli määramisel panna salt mingiks piisavalt juhuslikuks stringiks.
- Parooli kontrollimisel anda eelmine crypt() väljund ette salt argumendina, ning võrrelda *salt* ning crypt() tulemus omavahel. (kuna $ lõpetab, siis võib anda kogu tulemuse)

# Paroolide hoidmine

- Räsi ei toida enam hästi: MD5 puhul 5600 miljonit katset sekundis...
- Parooliräsi ei pea arvutama ühe korra; täiesti mõistlik on seda teha näiteks 100 korda
- See segab graafikakaardi abil ründajat oluliselt
- Ära leiuta midagi, võta bcrypt teek
-

# Stack smashing

- Kanaarilind (Canary)
  - Ubuntus vaikimisi, aga mujal väga mitte
- Aadressiruumi juhuslikkus (ASLR)

# Standardteekide probleemid

- Peamiselt stringide ja sisendi pikkuse kontrolli puudumine

# Malloc

- Topeltvabastus free() puhul probleemne
- Saab kontrollida MALLOC_CHECK 2 keskkonnamuutuja väärtuse abil näiteks
- Pärast vabastamist makroga pointer ka NULLida

# Mittenegatiivne arv

- Kasutada unsigned andmetüüpi

# Kompileerimisel

- gcc -Wall -Wpointer-arith -Wstrict-prototypes -O2

# Paranoilisust!